Feature-based Signal Selection for Post-silicon Debug using Machine Learning

***Abstract*—A key challenge of post-silicon validation methodology is to select a limited number of trace signals that are effective during post-silicon debug. Structural analysis used by traditional signal selection techniques are fast but lead to poor restoration quality. In contrast, simulation-based selection techniques provide superior restorability but incur significant computation overhead. While early work on machine learning based signal selection is promising [10], it is still not applicable on large industrial designs since it needs thousands of simulations of large and complex designs. In this paper, we propose a signal selection technique that addresses the scalability issue of simulation-based techniques while maintaining a high restoration performance. The basic idea is to train a machine learning framework using a small set of circuits, and apply the trained model to the bigger circuit under test, without any need for simulating the large industry-scale designs. This paper makes two fundamental contributions: i) this is the first attempt to show that learning from small related circuits can be useful for signal selection, and ii) this is the first automated signal selection approach that is applicable on industrial designs without sacrificing restoration quality. Experimental results indicate that our approach can improve restorability by up to 135.4% (8.8% on average) while significantly reduce (up to 37X, 16.6X on average) the runtime compared to existing signal selection approaches.**

I. Introduction

The goal of post-silicon validation is to ensure that the fabricated, pre-production silicon functions correctly while running actual applications under on-field operating conditions. Post-silicon validation is a complicated process that needs to be done under aggressive time to market schedules. It accounts for more than 50% of the total validation cost of the integrated circuit [9]. A key challenge in post-silicon debug process is *limited observability*: limitations of the number of output pins, along with the limitations imposed by power and area constraints on trace buffer size, imply that only a few hundreds (out of billions) of internal signals can be traced during the execution. Furthermore, the trace signals need to be selected during design phase with appropriate routing hardware to trace buffer in-place. With all these constraints, it is crucial to develop techniques to select a set of signals that maximizes the observability during post-silicon debug.

To address the discussed issues, there has been significant research on developing automated signal selection techniques through pre-silicon analysis of the design in RTL or gate level. The goal is to to select a set of trace signals S that maximizes the restorability during the debug process. Restoration is the process of reconstructing the unknown signals based on the observed trace signals. There are two main categories of signal selection techniques, structure-based and simulation-based. Structure-based techniques try to define a metric for the signals

TABLE I
Time complexity comparison of our approach and existing signal selection techniques. N is the number of flip-flops in the circuit under test.

Technique	**Mock Simulations**	**Other Computations**
Simulation-based [2]	$O(N^2)$	None
Hybrid [2]	$O(N)$	Fast metric evaluations
Learning-based [10]	$O(N)$	Fast model training and predictions
Our approach	None	Fast model training and predictions

based on the circuit structure which is used for evaluating the candidates - usually greedy heuristic [7], [2], [4]. These approaches are fast but provide inferior restoration performance. On the other hand, simulation-based techniques [3] use mock simulation/restoration process to identify the top candidates. They provide superior restoration performance but introduce scalability challenges for large circuits. A hybrid selection approach [6] has been proposed which tries to make a trade-off between metric-based and simulation-based approaches. However, use of less simulations to identify the top signals impacts the restoration performance of the final selected set of trace signals. Recently, a learning-based approach [10] has been proposed where it applies machine learning regression techniques to the circuit under test to reduce the overhead of $O(N^2)$ simulations by $O(N^2)$ fast predictions, where N is the number of flip-flops in the design. However, it still needs $O(N)$ simulations, typically thousands or millions of simulations of the design, to train the selection model, which limits its applicability on large industry scale circuits.

The main contribution of this paper is a novel technique for signal selection that retains or improves the restoration quality of simulation-based techniques while drastically reduces the signal selection time. To the best of our knowledge, our approach is the first attempt in creating an automated signal selection technique that is applicable on large industry-scale designs while providing the best possible restoration performance. Our approach is characterized by three key components: (1) simulation-based techniques are applied on a set of few small training circuits; and (2) a proper feature vector is created to apply machine learning techniques to learn the criteria for good trace signals; and (3) apply the model to the larger circuit under test. The main idea is to run only a small number of simulations to train the signal selection model using a set of small related circuits (one time process).

Subsequently, our approach will utilize fast predictions of the selection model replacing the need for expensive simulation runs during the selection process in the larger circuits. Table I summarizes the time complexity of our approach compared to the existing state-of-the-art signal selection techniques for a circuit with N flip-flops.

The remainder of the paper is organized as follows. Section II presents the relevant background. We describe our approach in Section III followed by experimental results in Section IV. We discuss the related work in Section V. We conclude the paper in Section VI.

II. Background and Motivation

Supervised learning is a technique of inferring an unknown output for an input vector using a set of training examples consisting of pairs of input vectors and corresponding known outputs. There is two main categories for supervised learning in machine learning, classification and regression. Classification is predicting whether an element belongs to a set of discrete values (or classes) whereas, regression is predicting the value for a continuous function. An example for classification is classifying an email as spam or legitimate email based on some features of its content and meta data (feature vector). On the other side, predicting the price of a house based on its features like location, size, and age (feature vector) would be a regression prediction as the price is a continues value. In fact, classification is a special case of regression where each class is assigned to a range of values (or probabilities) of the prediction function. This is illustrated in Figure 1. The classification predictor divides the outputs to different classes, square and star. On the other hand, the regression is a continuous function trying to fit the best line passing through the inputs and outputs.

Rahmani et al. [10] proposed an approach that utilizes regression techniques to reduce the number of simulations from O(N^2) in Chatterjee *et al.* [3] to O(N). However, it is still computationally prohibitive in large industry circuits. The feature-based signal selection approach we propose in this paper addresses this issue. We first create a selection regression model by applying simulation-based techniques to a set of small circuits. After that, our approach replaces mock simulations on the circuit under test with much faster prediction using the regression model. This makes our approach scalable and applicable to large industry circuits.

III. Feature-based Signal Selection

Figure 2 shows the overview of our approach and its relation to existing simulation based approaches [3], [6]. First, we choose a set of small training circuits to build the selection model. For each training circuit, we apply a modified version of both elimination-based [3] and augmentation-based [6] approaches and select the best result. We then generate a set of training vectors and add it to the training vectors set. Next, we use this training set to create a selection model using different machine learning regression techniques and pick the one with best accuracy. In this step, we train a model that learns the criteria of a good candidate signal and the relation with its properties. This model can then be used to select trace signals on any related design under signal selection without expensive mock simulations involved. It should be noted that the selection model training is a one time process. Once it is done the model can be used to select trace signal on any other related circuits, as long as it shares same properties like connectivity and coding guideline with the training circuits.

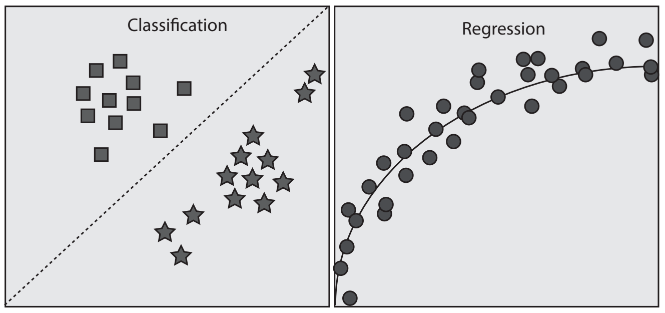
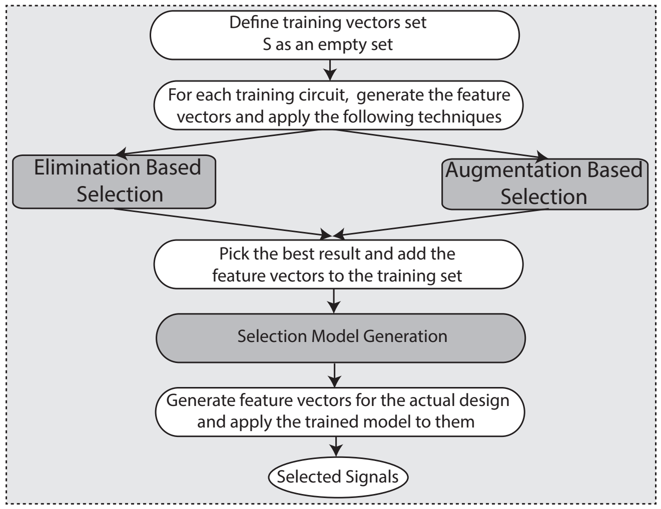


Fig. 1. Two different type of supervised learning. Classification is predicting whether an element belongs to a set of discrete values (or classes) whereas, regression is predicting the value for a continuous function.


Define training vectors set S as an empty set
For each training circuit, generate the feature vectors and apply the following techniques
Elimination Based Selection
Augmentation Based Selection
Pick the best result and add the feature vectors to the training set
Selection Model Generation
Generate feature vectors for the actual design and apply the trained model to them
Selected Signals
Fig. 2. Overview of our proposed approach and its relation to existing simulation based approaches [3], [6]. We generate a set of training vectors by applying simulation-based techniques to a set of small training circuits. We then use this training set to create a selection model which can be used to select trace signals on any design without expensive mock simulations involved.

A. Problem Formulation

The goal of any signal selection algorithm is to select a set of trace signals S which includes w signals (out of N signals in the circuit), so that the restored signals during the post-silicon debug process is maximized. Here w, which is the trace buffer width, is a parameter of the selection algorithm. To motivate our discussion in the next sections, we provide a formulation of signal selection as a constrained optimization representation. Note that any candidate signals set S can be mapped to a unique *input vector* $v = \langle f_1, f_2, ..., f_N \rangle$ where $f_i \in \{0, 1\}$. Informally, $f_i = 1$ means i-th flip-flop is selected and is present in S. On the other hand, $f_i = 0$ means that i-th flip-flip is not selected as part of the signals in S. This means, input vector v completely identifies the candidate signals set S and vice versa. We will refer to v as the *candidate vector* of S and S as the *candidate signal set* of v. In addition, we define $r_m(v)$ to be the number of signals that can restored where signals in v are traced in a windows of m cycles. Given that, we can formulate signal selection as the optimization problem defined below.

$$\text{maximize } r_m(v)$$
$$\text{under constraint } \sum_{k=1}^{N} f_k = w \tag{1}$$

The above optimization problem includes both the simulation windows (m) as well as the trace buffer width (w) as the parameters.

B. Elimination-based Signal Selection

Algorithm 1 outlines the steps involved in the elimination-based signal selection approach in Chatterjee *et al.* [3]. First, all the flip-flops are selected as part of the candidate signals set (*i.e.*, they are set to 1 in v). In each iteration of the algorithm, a signal with minimum impact on the restoration ratio of the candidate signals vector is removed by setting its value to 0 in v. This process stops when the number of remaining flip-flips in the candidate signals vector is equal to the trace buffer width (w). The final selected signals in v is returned as the output for the algorithm. It should be noted that our approach is not completely identical to Chatterjee *et al.* [3] as we do not run any pre-processing on the initial vector v for coarse-grained pruning of the signals which can significantly degrade the performance of the final set of signals. Our approach does not have computation limitation similar to [3] as we run this algorithm only on a set of small training circuits.

C. Augmentation-based Signal Selection

Algorithm 2 outlines the steps involved in selecting signals using the augmentation-based technique similar to the approach described by Li *et al.* [6]. In this technique, in each iteration, we add the most beneficial signal to the candidate signals set, instead of removing the least beneficial one. This process stops when the total number of selected signals is equal to trace buffer width w. The final vector of selected signals v is returned as the algorithm output. Our approach is different from Li *et al.* [6]. Because of computational limitation, they use mock simulations only for top 5% of the candidate signals, which can degrade the restoration performance of the final selected signals. However, we do not have this limitation as we run this selection technique only on a set of small training circuits, not the actual circuit under test.

D. Selection Model Generation

The core part of our approach is the training model generation and how to best choose the feature vectors. The model should be generic enough so that it can be applied to the circuit under test and accurate enough to produce high quality result. Before going into details of our proposed modeling technique, we would like to define few terms and functions for a circuit with N flip-flops, mock simulation window m.

Algorithm 1 Elimination-based Signal Selection

```
1:  procedure EliminationBased($circuit, w, m$)
2:      Create initial vector of $v =< 1, 1, ..., 1 >, |v| = N$
3:      $remainedSignals = N$
4:      while $remainedSignals > w$ do
5:          $maxRestorability = -\infty$
6:          $maxIndex = -1$
7:          for $i = 1; i <= N; i++$ do
8:              if $v[i] = 1$ then
9:                  $v[i] = 0$
10:                 if $r_m(v) > maxRestorability$ then
11:                     $maxRestorability = r_m(v)$
12:                     $maxIndex = i$
13:                 end if
14:                 $v[i] = 1$
15:             end if
16:         end for
17:         $v[maxIndex] = 0$
18:         $remainedSignals = remainedSignals - 1$
19:     end while
20:     return v
21: end procedure
```

Algorithm 2 Augmentation-based Signal Selection

```
1:  procedure AugmentationBased($circuit, w, m$)
2:      Create initial vector of $v =< 0, 0, ..., 0 >, |v| = N$
3:      for $selected = 1; selected <= w; selected++$ do
4:          $maxRestorability = -\infty$
5:          $maxIndex = -1$
6:          for $i = 1; i <= N; i++$ do
7:              if $v[i] = 0$ then
8:                  $v[i] = 1$
9:                  if $r_m(v) > maxRestorability$ then
10:                     $maxRestorability = r_m(v)$
11:                     $maxIndex = i$
12:                 end if
13:                 $v[i] = 0$
14:             end if
15:         end for
16:         $v[maxIndex] = 1$
17:     end for
18:     return v
19: end procedure
```

- **Fan-out$_g$(x)** for flip-flop f is defined as the number of gates of type g (and, or, etc.) connected to its output.
- **Fan-in$_g$(x)** for flip-flop f is defined as the number of gates of type g (and, or, etc.) connected to its inputs.
- **Connectivity(x)** for flip-flop f is defined as the number of flip-flops connected to it through other combinational gates in both backward and forward directions.

- **InputDistance(x)** for flip-flop f is defined as the minimum distance of the flip-flop from the primary input signals (in terms of number of gates).
- **OutputDistance(x)** for flip-flop f is defined as the minimum distance of the flip-flop from the primary output signals (in terms of number of gates).
- **ZeroProbability(x)** for flip-flop f is defined as the percentage of 0 values for the flip-flop in a mock simulation over m cycles.
- **SingleRestoration(x)** for flip-flop f is defined as the number of restored states in a mock simulation/restoration process over m cycle if f is the only trace signal.
- **SelectionOrder$_g$(x)** for flip-flop f is defined as the sequence number of selection when technique g is applied. This number is 1 for the first (best) selected signal and N for the last selected signal.
- **Rank(g(x))** for flip-flop f is defined as the number of flip-flops with $g(x) <= g(f)$ divided by N. In other words, it is the normalized relative rank of applying function g to flip-flop f compared to the other flip-flops. This value would be 1 and $1/N$ for the flip-flops with the maximum and minimum value of g(f), respectively.

1) Feature Selection: Selecting the right features is the most important part of any machine learning problem as it directly impacts the quality of the model and solution. In our case, the features should be selected such that it can model the true correlation between structural properties of a signal and its performance in state restoration. In addition, it should be independent of the circuit size and structure. This is crucial as we want to train our model using a set of small circuits and apply the learning to the bigger circuit under test. Lastly, the generation of feature vectors should not be computationally expensive so it can easily scale while selecting signals in large industry-scale circuits. In order to address these requirements, we define feature vector v for flip-flop f in circuit c to have the following components.

- Rank(Fan-out$_g$(f)) for all gates of type g in the circuit.
- Rank(Fan-in$_g$(f) for all gates of type g circuit.
- Rank(Connectivity(f)) for flip-flop f.
- Rank(InputDistance(f) for flip-flop f.
- Rank(OutputDistance(f)) for flip-flop f.
- Rank(ZeroProbability(f)) for flip-flop f.
- Rank(SingleRestoration(f)) for flip-flop f.

It can be seen that we have chosen features that are mostly based on the circuit structure and fast to evaluate. In addition, we are applying rank function to all the features to make them relative values instead of using the absolute values. This makes the features independent of the circuit size and number of gates. Intuitively, our feature vector captures the fan-in and fan-out of the signal, its relative position and depth in the circuit, and its impact on restoring its neighbors when selected as a trace signal. Our experiments have shown that there is a high correlation between these features and the restoration performance of a flip-flop.

2) Model Selection: In this step, we generate a selection model by applying simulation based techniques on a small set of training circuits. Intuitively, the model learns the criteria of a good trace signal and the relation with its feature vector described before. Algorithm 3 outlines the steps involved in creating our selection model from a set of small training circuits. For each circuit, we apply both augmentation and elimination based techniques and pick the one with better result (the one with better average restoration ratio for trace buffer widths 8, 16, and 32). Next, for each flip-flop f in the circuit we add a pair of feature vector v and selection order rank r to the training vectors set. Choosing selection order rank helps to normalize the training data across all the circuits and makes it independent of number of flip-flops in the circuit. We then apply different regression techniques (like svm, linear modeling, etc.) to the training vectors and return the best one as the result. To measure the quality of a model on a test vector set of size n, we use Mean Prediction Error (MPE) defined as below.

$$MeanPredictionError = 1/n * \sum_{k=1}^{n} |\hat{r}(v_k) - r(v_k)| \quad (2)$$

Where $r(v_k)$ is the actual value and $\hat{r}(v_k)$ is the predicted value of r for v_k. In order to avoid over-fitting [1] in our model training, we use 5-fold cross-validation technique where we use 20% of the vectors as test (validation) vectors and 80% as the the training vectors.

Algorithm 3 Model Generation Algorithm

```
1: procedure MODELGENERATION(trainingCircuits, regressionModels, m)
2:     Create training vectors set trainingVectors
3:     for each circuit c in trainingCircuits do
4:         n = number of flip-flops in c
5:         apply AugmentationBased(c,n,m) and EliminationBased(c,1,m) to c and pick the best one as g
6:         for each flip-flop f in c do
7:             Add $< v, r >$ to trainingVectors where v is the feature vector for the flip-flop and r is Rank(SelectionOrder$_g$(f))
8:         end for
9:     end for
10:    apply all the models in regressionModels to trainingVectors
11:    return model m with minimum MPE
12: end procedure
```

E. Signal Selection Process

Once we have our selection model trained, it can be used on any circuit for selecting trace signals. To motivate our approach, we have used the smallest circuits in ISCAS'89 benchmarks [2] to train our model using *cubist* regression technique. Figure 3 shows the actual versus predicted selection ranks (using the trained model) on a set of flip-flops in s38584 benchmarks (in this example, s38584 is assumed as the actual design under signal selection). It can be seen that there is a high correlation between the real and predicted values. This is significant as it enables us to select high quality trace signals in the circuit

[1]Over-fitting happens when the model is too specific to the training data, resulting in a high accuracy in training data and low accuracy in new data.

[2] s1494, s1488, s713, s1238, s1196, and s838.

under test using very fast predication to generate the selection ranks based on the feature vectors, instead of expensive mock simulations.

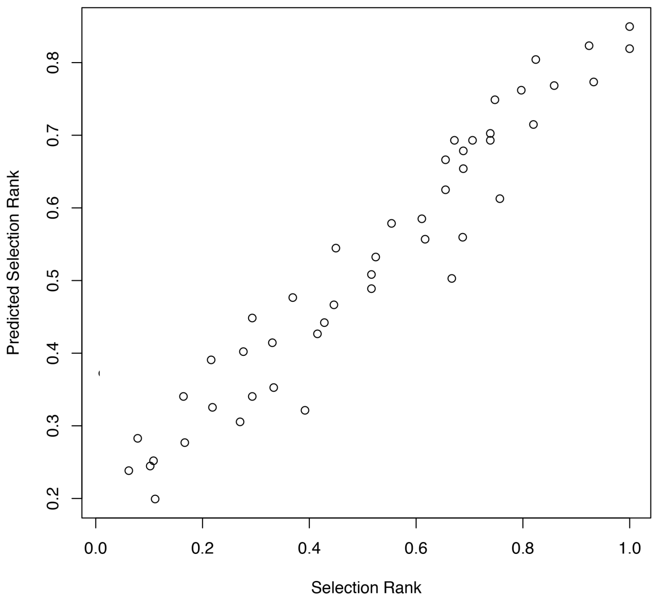


Fig. 3. Actual versus predicted values of selection ranks in s38584 benchmark. The high correlation between the values enables us to have high quality trace signal selection using fast predictions instead of mock simulations.

Algorithm 4 outlines our proposed signal selection technique. We first generate feature vectors for all the flip-flops in the circuit. We then use these vectors to predict the selection sequence rank for the flip-flops using the given selection model *m*. We return the top *w* (trace buffer width) flip-flops with the highest value of predicted selection sequence rank as the result.

Algorithm 4 Signal Selection Algorithm

```
1: procedure SIGNALSELECTION(circuit, m, w)
2:     Initialize predictionMap as an empty map
3:     for each flip-flop f in circuit do
4:         v = feature vector of f
5:         r = m(v), the predicted value of selection sequence
           rank of f using model m
6:         Add $< f, r >$ to predictionMap
7:     end for
8:     Sort predictionMap based on r values
9:     return top w flip-flops with the highest values of r
10: end procedure
```

IV. Experiments

A. Experimental Setup

In order to evaluate the effectiveness of our proposed signal selection technique, we have developed a cycle-accurate simulator for ITC'99 and ISCAS'89 benchmarks suites in C++. In addition to regular simulation, our simulator can also perform restoration process using both forward and backward restoration rules. Our simulator iterates on the unknown signal and attempts to apply forward and backward restoration rules to restore their values. The process stops when it is not possible to restore any more signals in the iteration. We validated the correctness of our simulator by comparing the output values with the output of Verilog simulation of the same benchmarks using *Icarus Verilog* [11] simulator. We used the set of largest circuits in ISCAS'89 as has been studied by previous works. We also used the largest circuits in ITC'99 benchmarks. We used a set of regression modeling techniques (glm, svmLinear, cubist, earth, gaussprLinear, svmPoly, and treebag) in *caret* package in *R* [8] as the modeling/prediction tool. In addition, while running the modeling techniques, we used *5-fold cross validation*. We used a set of smallest ISCAS'89 benchmarks (s1494, s1488, s713, s1238, s1196, and s838) to train our selection model.

In our experiments, we did not use the reported numbers of Chatterjee *et al.* [3] and Li *et al.* [6], as they used modified versions of ISCAS'89 benchmarks with some specific optimizations applied. In order to have a fair comparison, we tried to obtain the executables of their implementations. Li et al. [6] provided us with the executable file of their signal selection implementation and we used it for the selection process. Unfortunately, we were not able to get the implementation of Chatterjee *et al.* [3]. We used our own implementation of their approach in our set of experiments. However, we used the same parameters $c = 64$ and $PT = 95\%$ as they reported in their paper. In addition, we used $m = 32$ as our mock simulation window. In order to calculate the restoration ratios, we fed our simulator with 100 sets of random input vectors and reported the average restoration ratios for the selected set of signals. However, we forced the circuits to run in their normal operating mode by fixing their relevant control (reset) signals, while randomly assigning values to all the other inputs. The control signals include active low reset signals *g35* in *s38584* and *RESET* in *s35932* which were set to '1' in our experiments.

B. Model Selection

In order to choose the best regression model for our signal selection application, we explored several models available in *caret* package [8] in *R*. Figure 4 illustrates real versus estimated values of selection rank in S38584 benchmark using training circuits set of s1494, s1488, s713, s1238, s1196, and s838. It can be observed that *cubist* is the best model in our experiments with minimum prediction error and highest correlation between the real and estimated value. In fact, *cubist* outperformed other models consistently for other benchmarks as well. For this experiment, we used 80% of our training vector for actual training and the other 20% for the testing. This can prevent us from biasing while training the models. We selected *cubist* model as our regression model for the rest of our experiments. *Cubist* is a non-linear model, simpler than neural network, and designed to analyze millions of records which makes it a good fit for our large scale application [1].

TABLE II

RESTORATION RATIOS USING OUR APPROACH COMPARED WITH EXISTING SELECTION APPROACHES.

Circuit	#Flip-flops	Buffer Width	Simulation-based [3]	Hybrid [6]	Learning-based [10]	Our Approach	Imp. over the best
s5378	179	8	13.41	**14.35**	14.20	14.13	-1.5%
		16	7.35	8.36	**8.40**	8.92	6.1%
		32	4.47	**4.99**	4.93	5.12	2.6%
s9234	228	8.0	13.98	9.25	**15.33**	15.82	3.2%
		16	8.30	6.13	**8.76**	9.10	3.9%
		32	4.46	4.38	**4.84**	5.11	5.6%
s15850	597	8	26.33	21.90	**44.03**	45.12	2.5%
		16	19.89	14.78	**23.13**	24.37	5.4%
		32	13.19	10.88	**13.92**	13.82	-0.7%
s13207	669	8	35.52	33.60	**47.18**	49.30	4.5%
		16	20.13	23.22	**29.00**	31.21	7.6%
		32	11.25	13.64	**15.42**	16.13	4.6%
s38584	1452	8	N/A	27.00	**54.25**	127.72	135.4%
		16	N/A	13.97	**69.03**	79.09	14.6%
		32	N/A	7.50	**43.66**	44.02	0.8%
s38417	1636	8	N/A	37.71	**52.33**	53.27	1.8%
		16	N/A	23.80	**27.12**	26.97	-0.5%
		32	N/A	11.83	**16.73**	17.10	2.2%
s35932	1728	8	132.00	144.00	**186.80**	186.90	0.1%
		16	67.45	72.00	**93.60**	93.42	-0.1%
		32	34.63	36.00	**46.98**	47.15	0.4%
b15	449	8	5.99	N/A	**6.15**	7.18	16.7%
		16	3.56	N/A	**4.83**	4.98	3.1%
		32	34.63	N/A	**3.31**	3.46	4.53%
b17	1415	8	N/A	N/A	**14.12**	14.43	2.1%
		16	N/A	N/A	**13.19**	13.31	0.9%
		32	N/A	N/A	**7.93**	8.77	10.6%
b18	3320	8	N/A	N/A	N/A	25.12	N/A
		16	N/A	N/A	N/A	21.60	N/A
		32	N/A	N/A	N/A	12.49	N/A
b19	6642	8	N/A	N/A	N/A	32.00	N/A
		16	N/A	N/A	N/A	24.64	N/A
		32	N/A	N/A	N/A	18.11	N/A

C. Restoration Quality

Table II presents the restoration ratios of our approach compared with previous state-of-the-art techniques [3], [6], [10] using different ISCAS'89 and ITC'99 benchmarks. The trace buffer sizes used in our experiment are $8 \times 4k$, $16 \times 4k$, and $32 \times 4k$. The corresponding restoration ratio for each technique is reported. The ones shown as 'N/A' for [3] and [10] means that their technique was not able to finish within 24 hour of runtime. In addition, unfortunately we were not able to get the result of [6] for ITC'99 benchmarks as their binary failed to parse the benchmarks. The last column indicates the percentage of improvement using our approach compared with the best (shown in bold) result provided by the existing approaches. The results indicate that our approach consistently performs comparable or better compared to existing approaches. The improvement in restoration performance is up to 135.4% in *s38584* and 8.8% on average. Compared to Chatterjee *et al.* [3], we run the elimination-based technique on training circuits without any pruning which reduces the chance of removing effective flip-flops prior to selection itself. Similarly, Li *et al.* [6] incorporated simulations for only top 5% of the candidate flip-flops, which sacrifices the precision of the selection process. In addition, building the selection model using small training circuits, allows us to run both elimination-based and augmentation-based techniques at the same time and pick the best one for each circuit. Compared to Rahmani *et al.* [10], we train the selection model based on the training circuits structure and the best result of simulation-based techniques, whereas they use machine-learning to just reduce the number of mock simulation on the circuit under test. In addition, our model is trained using the best result of simulation-based techniques on a set of training circuits (instead of just one), which provides a more globally optimized selection model. Finally, although our model is trained using small circuits in ISCAS'89 benchmarks, it still outperforms [10] in ITC'99 benchmarks (b15 and b17). This shows that the proposed feature vector and selection model is generic enough that can be applied to the designs from same domain with similar characteristic and standard cell library. This enables training the model with a small set of related circuit with same coding guideline or sub-components of SoCs and use it for signal selection in the larger industrial circuits.

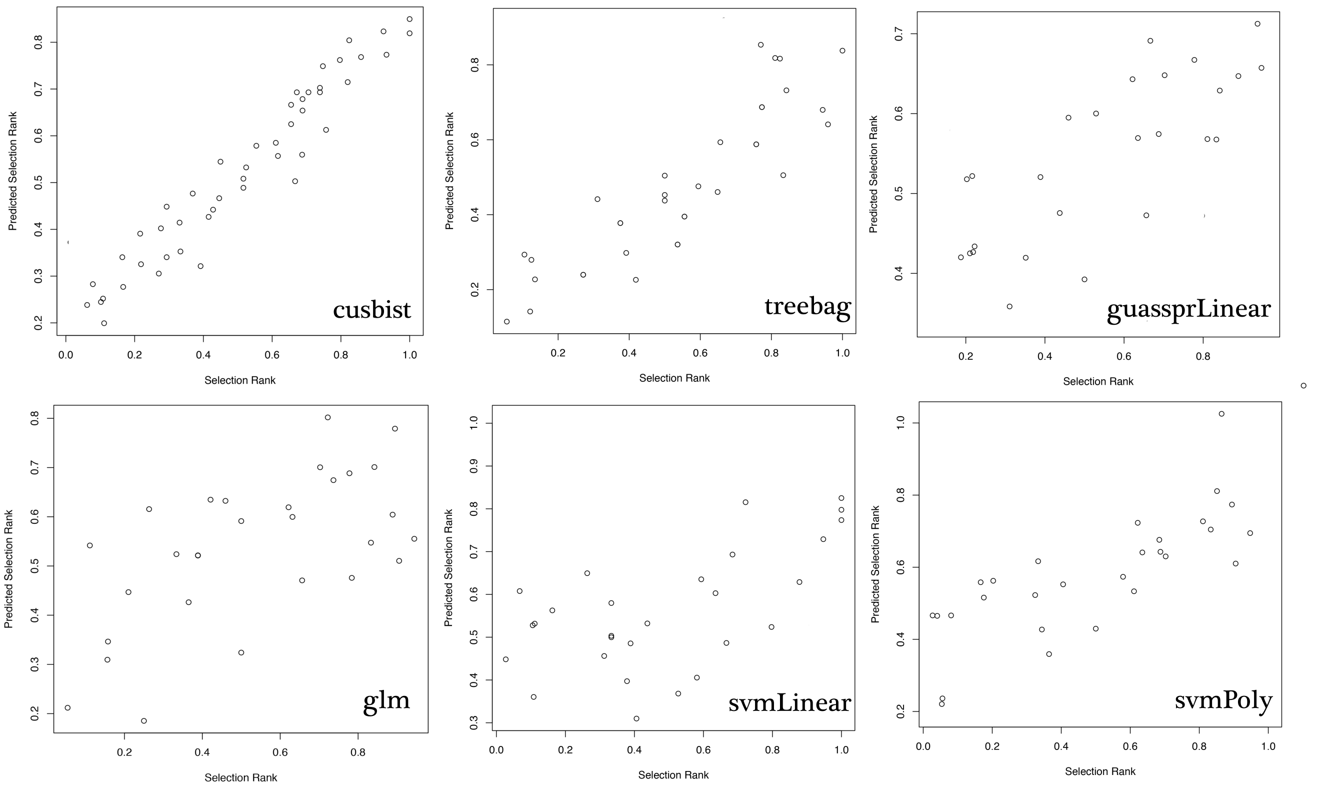


Fig. 4. Real versus estimated values of selection rank in S38584 benchmark using training circuits set of s1494, s1488, s713, s1238, s1196, and s838.

D. Selection Time and Scalability

To compare the runtime of different approaches, we used an Octa-Core AMD Opteron 6378 (1400 MHz) machine with 188GB of memory for all the experiments. The runtime for our approach is calculated as the summation of required time for generating training vectors on the training circuits, modeling, generating the feature vectors for the circuit under test, and the signal selection process itself. Table III presents the runtime of our approach compared with the previous techniques [3], [6], [10] using different ISCAS'89 and ITC'99 benchmarks. The reported runtime format is 'hour:minute:second'. The ones shown as 'N/A' for [3] and [10] means that their technique was not able to finish within 24 hour of runtime. In addition, unfortunately we were not able to get the result of [6] for ITC'99 benchmarks as their binary failed to parse the benchmarks. As expected, our approach is significantly faster than the existing approaches. The speed-up is up to 37X in *s38417* and *b17* for buffer width of 32 and 17.6X on average. This is because in our approach we need mock simulations only on a set of small training circuits. Once the model is created, there is no need for any simulations on the circuit under test as the selection process just uses fast predictions instead of actual simulations. This makes our approach significantly more scalable and practical compared to the existing ones. In summary, our approach not only produces comparable or better restoration quality, but it is also significantly faster than the existing approaches.

V. Related Work

Limited observability of the internal signals is the primary challenge in post-silicon validation debug process. Trace buffers provide one of the commonest form of on-chip instrumentations. The primary challenge with trace buffers is to compute *a priori* a small set of signals that can be traced in order to maximize reconstruction of internal states. Ko et al. [4] and Liu et al. [7] have proposed efficient signal selection algorithms based on partial restorability. Basu et al. [2] improved their methods by proposing an efficient algorithm that selects signals based on their total restorability.

Existing signal selection approaches can be classified in two categories, *structural-based* and *simulation-based*. Approaches in the first category, use some sort of greedy heuristic algorithms to iteratively select signals to optimize a metric based on the circuit structure [4], [7], [2]. They are relatively efficient in computation speed, but have poor restoration quality compared to simulation-based algorithms. Simulation-based algorithms are based on the intuition that if a set of signals works well for some random input vectors then it is also likely to provide high state reconstruction on other inputs and therefore a high restorability ratio. Chatterjee *et al.* [3] demonstrated that simulation-based signal selection is a promising approach. However, their approach requires O(N^2) simulations where N is the number of flip-flops in the circuit. This makes their approach computationally expensive for large circuits. Li *et al.* [6] proposed a hybrid (metric-based and simulation-based) signal selection technique. However, to save selection time,

TABLE III
RUNTIME COMPARISON OF OUR APPROACH COMPARED WITH EXISTING SELECTION APPROACHES.

Circuit	Buffer Width	Simulation-based [3]	Hybrid [6]	Learning-based [10]	Our Approach	Speedup
s5378	8	00:01:53	00:00:08	00:01:46	00:00:11	0.7X
	16	00:01:52	00:00:10	00:01:52	00:00:11	0.9X
	32	00:01:48	00:00:16	00:02:09	00:00:11	1.5X
s9234	8	00:08:52	00:00:32	00:00:10	00:00:11	1.0X
	16	00:08:43	00:00:40	00:00:10	00:00:11	1.0X
	32	00:08:10	00:00:50	00:00:10	00:00:11	1.0X
s15850	8	03:44:12	00:05:20	00:04:20	00:00:1	20.1X
	16	03:44:04	00:06:00	00:04:35	00:00:13	21.2X
	32	03:43:39	00:06:36	00:05:04	00:00:13	23.4X
s13207	8	01:21:41	00:01:36	00:03:45	00:00:13	7.4X
	16	01:21:35	00:02:00	00:04:01	00:00:13	9.2X
	32	01:21:13	00:02:40	00:04:12	00:00:13	12.3X
s38584	8	N/A	00:05:28	00:16:52	00:00:36	9.1X
	16	N/A	00:06:06	00:17:09	00:00:36	10.2X
	32	N/A	00:09:02	00:17:35	00:00:36	15.1X
s38417	8	N/A	00:22:42	00:20:23	00:00:39	31.4X
	16	N/A	00:33:04	00:21:07	00:00:39	32.5X
	32	N/A	00:34:28	00:23:55	00:00:39	36.8X
s35932	8	11:39:36	00:04:28	00:16:49	00:00:37	7.2X
	16	11:39:09	00:05:56	00:17:33	00:00:37	9.6X
	32	11:38:01	00:08:38	00:18:21	00:00:37	14.0X
b15	8	06:12:09	N/A	00:06:49	00:00:12	34.1X
	16	06:09:55	N/A	00:07:03	00:00:12	35.3X
	32	06:06:40	N/A	00:07:11	00:00:12	35.9X
b17	8	N/A	N/A	00:19:10	00:00:35	32.9X
	16	N/A	N/A	00:20:30	00:00:35	35.1X
	32	N/A	N/A	00:21:40	00:00:35	37.1X
b18	8	N/A	N/A	N/A	00:06:11	N/A
	16	N/A	N/A	N/A	00:06:11	N/A
	32	N/A	N/A	N/A	00:06:11	N/A
b19	8	N/A	N/A	N/A	00:21:09	N/A
	16	N/A	N/A	N/A	00:21:09	N/A
	32	N/A	N/A	N/A	00:21:09	N/A

[6] uses simulation for a small fraction of the signals and thereby sacrifices restoration performance. A learning-based approach [10] has been proposed, where it applies a learning technique to the circuit under test to reduce the simulations overhead. However, it still needs O(N) simulations on the actual circuit under test to train the model which can be a limiting factor in large industry scale chips. Our work is the first attempt that utilizes machine learning techniques to learn selection criteria from a set of small circuits and apply it on the bigger circuit under test without any expensive simulation involved.

VI. CONCLUSIONS

Post-silicon validation and debug is an expensive and important phase in integrated circuit design flow. The success for post-silicon debug depends on the quality of trace signals that can maximize the effective use of limited observability in trace buffer. Therefore, it is crucial to develop effective signal selection techniques that can provide high restoration performance and can be applied on large industrial designs. Existing state-of-the-art signal selection techniques yield signals with good restorability, but are computationally prohibitive for industrial designs. We presented a learning-based signal selection approach which provides comparable or better restorability while providing an order-of-magnitude reduction in signal selection time, making it applicable for industry-scale circuits.

REFERENCES

[1] Cubist Regression Model. https://www.rulequest.com/cubist-info.html.
[2] K. Basu and P. Mishra. Rats: Restoration-aware trace signal selection for post-silicon validation. IEEE TVLSI, 2013.
[3] D. Chatterjee et al. Simulation-based signal selection for state restoration in silicon debug. ICCAD, 2011.
[4] H. F. Ko and N. Nicolici. Algorithms for state restoration and trace-signal selection for data acquisition in silicon debug. IEEE TCAD, 2009.
[5] H. F. Ko and N. Nicolici. Combining scan and trace buffers for enhancing real-time observability in post-silicon debugging. ETS, 2010.
[6] M. Li and A. Davoodi. A hybrid approach for fast and accurate trace signal selection for post-silicon debug. DATE, 2013.
[7] X. Liu and Q. Xu. Trace signal selection for visibility enhancement in post-silicon validation. DATE, 2009.
[8] Max Kuhn. The caret Package. http://topepo.github.io/caret/index.html.
[9] A. Nahir et al. Bridging pre-silicon verification and post-silicon validation. DAC, 2010.
[10] K. Rahmani et al. Postsilicon trace signal selection using machine learning techniques. IEEE TVLSI, 2016.
[11] Stephen Williams. Icarus Verilog. http://iverilog.icarus.com/.